



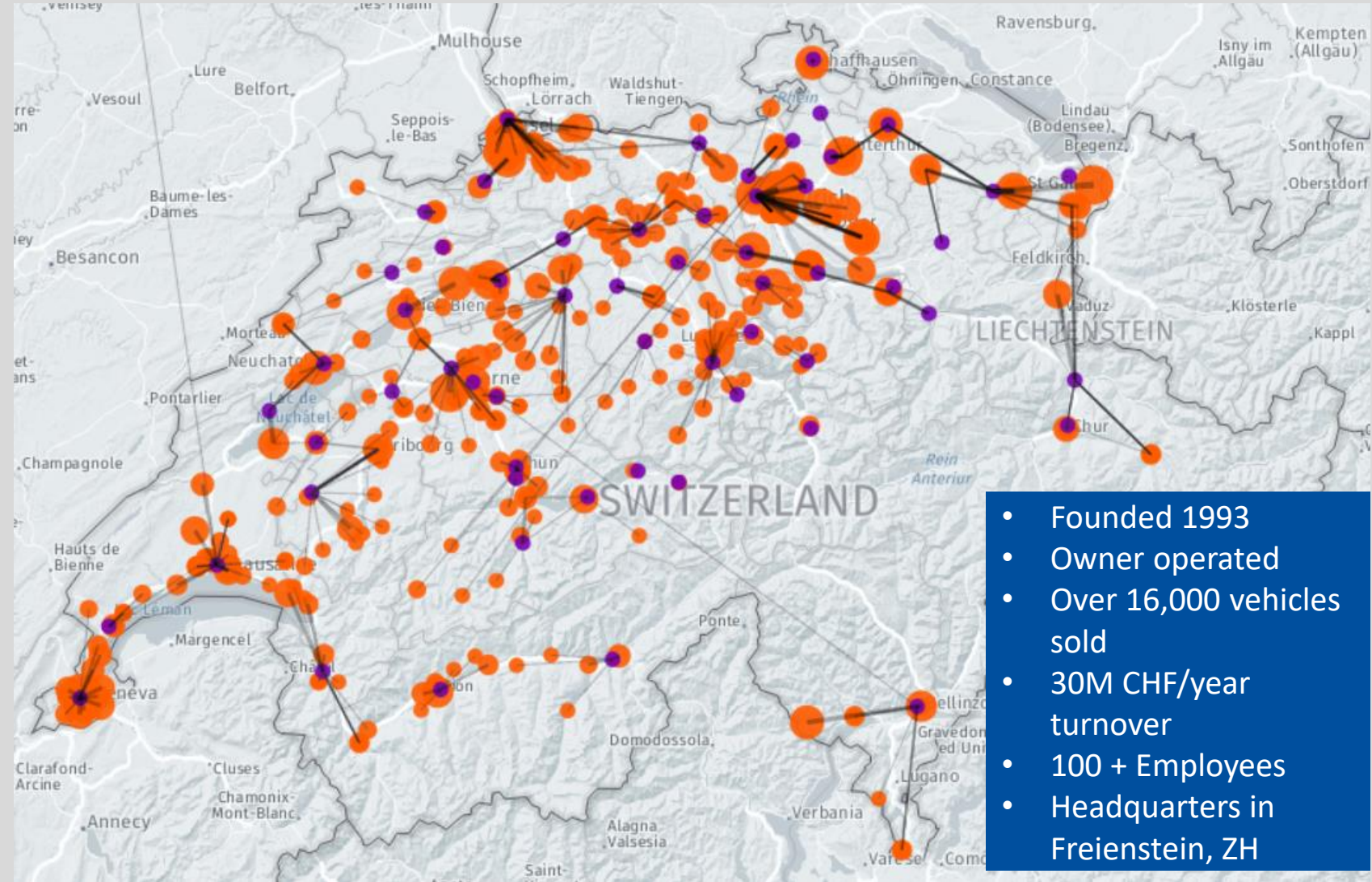
Courtesy of ETH-Zurich, ARC

Designing and controlling safe self-driving systems

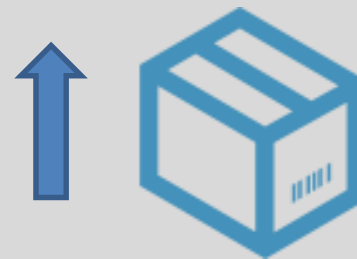
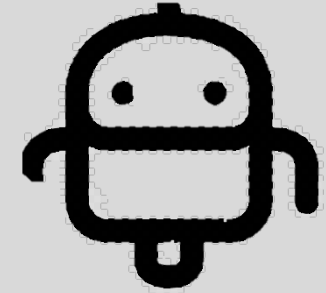
Dr. Erik Wilhelm
Head of Research
KYBURZ Switzerland

23rd May, 2019

A well-established brand



Changing postal delivery landscape



- Must be:
 - Cheaper
 - Faster
 - More reliable
 - ... More personal?

Prototype series



- Mobile depot box (eT2)
- Sensors
 - 2D Lidar
 - Ultrasonic
 - 360 camera
 - GPS
 - Bump-stop



- Autonomous delivery agent (eT3)
- Sensors
 - 3D Lidar
 - Ultrasonic
 - Infrared
 - INS
 - Bump-stop



- Flexible delivery system (eT4)
- Sensors
 - 3D Lidar (2x)
 - Ultrasonic (8x)
 - Infrared (8x)
 - Radar (4x)
 - GPS (INS)
 - 360 Cameras (localization)
 - 360 Cameras (comprehension)
 - Time-of-flight camera
 - Bump-stop

Autonomous System Design Challenges

High availability



Image: ABC news

Ap(proved) safety

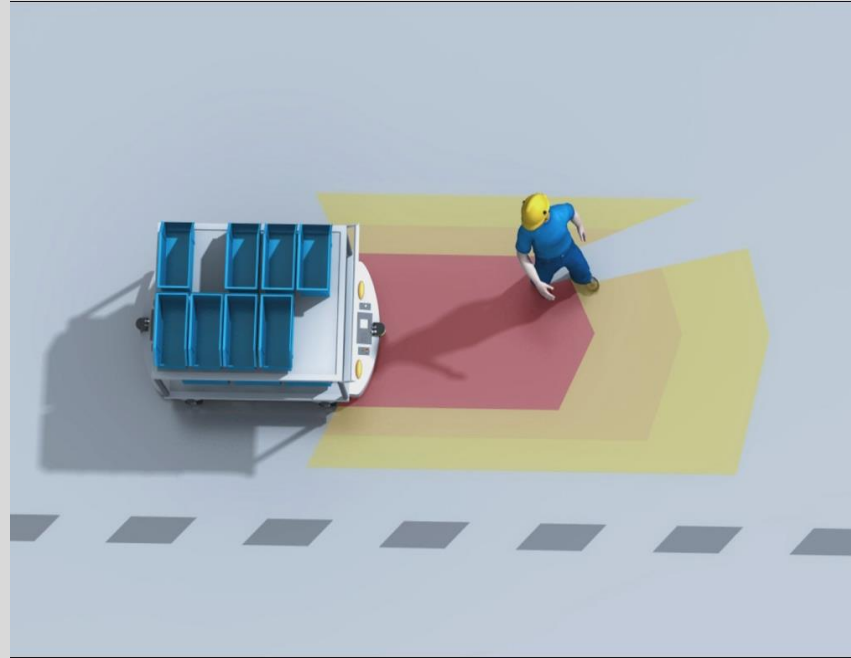


Image: sick.com

Test coverage



Image: youtube.com

Availability Requirement



Image: Frugal Entrepreneur

- 300 parcels/day
- 8.25 hr/day
- 56 kCHF/year



- 40 parcels/day
- 24 hr/day
- 50 kCHF purchase





Image: cnbc.com

- 1 disengagement/day
- 56 hours per year
- 3 kCHF per year

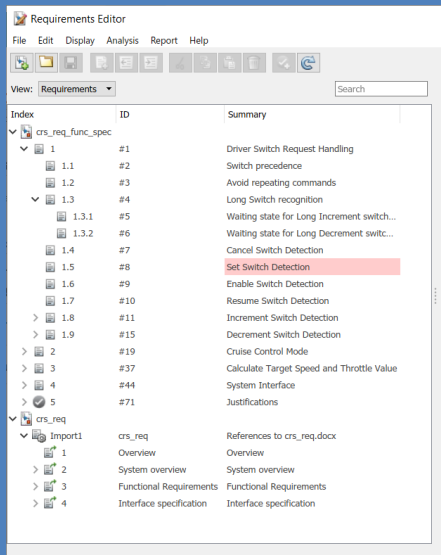
- Robotic delivery amortized with 1 disengagement/day, never with 3 disengagements/day

Sensor and controller redundancy

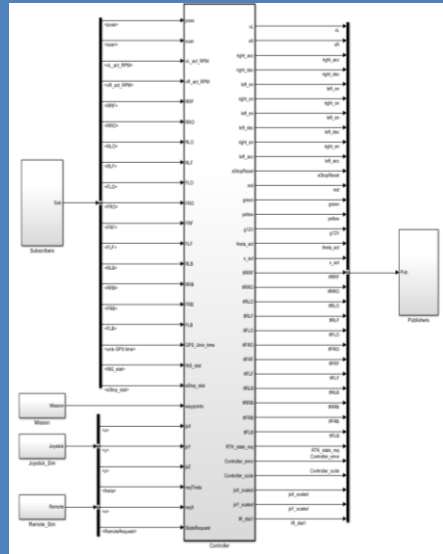
Localization	Day	Night	Precipitation	Fog	Tall structures	Tranparent obstacles	Diffuse obstacles
INS (GPS)	✓	✓	✓	✓	!		
Optical	✓	✗	✗	!	✓		
Pointcloud	✓	✓	!	✗	✓	-	-
Obstacle Avoidance							
LiDAR	✓	✓	!	✗	-	✗	!
Optical	✓	✗	✗	!	-	✓	✓
Radar	✓	✓	!	✓	-	✓	✓
Ultrasonic	✓	✓	✓	✓	-	✓	!
Time-of-Flight	!	✓	!	✗	-	!	✗
Infrared	✗	✓	!	✗	-	✗	✗
Bump Stop	✓	✓	✓	✓	-	✓	✓

Workflow

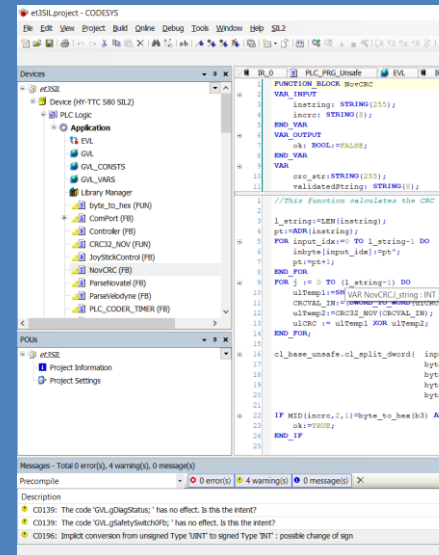
Specifications



Model Based Design



Code Generation

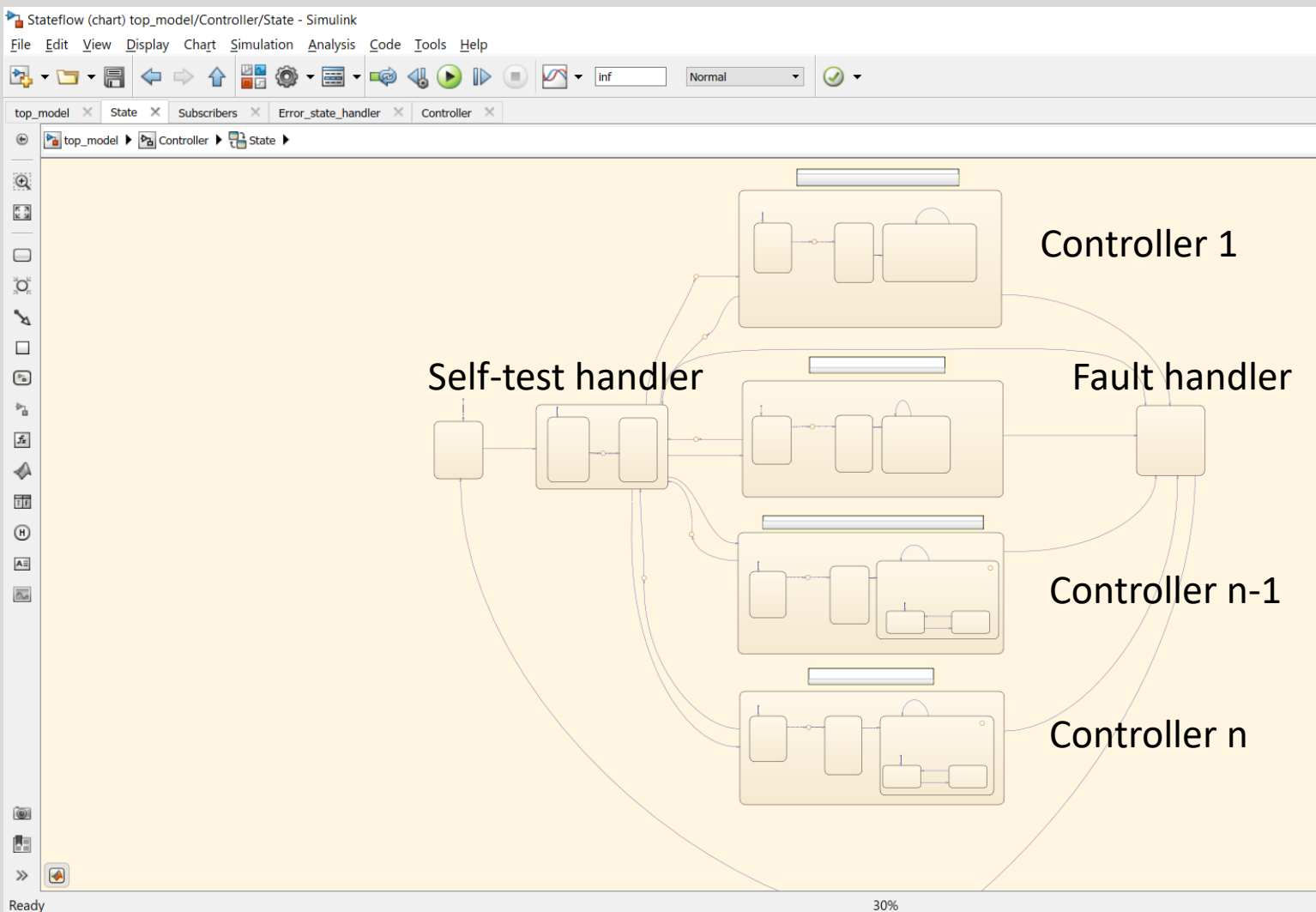


Compilation



- This workflow allows SIL2 certifiable code to be generated using model-based design
- Review and testing occurs within each phase and before each release

Availability Solution



- Supervisory controller invokes multiple independent and redundant motion control paradigms
 - Local
 - Remote
 - Mission training
 - Mission running
- Graphical state modeling of control logic allows streamlined, debuggable, testable strategies

Functional Safety and Approvals

- Kyburz is designing autonomous machines not vehicles
 - IEC 61508
- Voluntarily following automotive functional safety norms
 - ISO 13849:2015
 - ISO 26262:2018
- Primary implications
 - Development process
 - Documentation system
 - Component selection
 - Software development toolchains

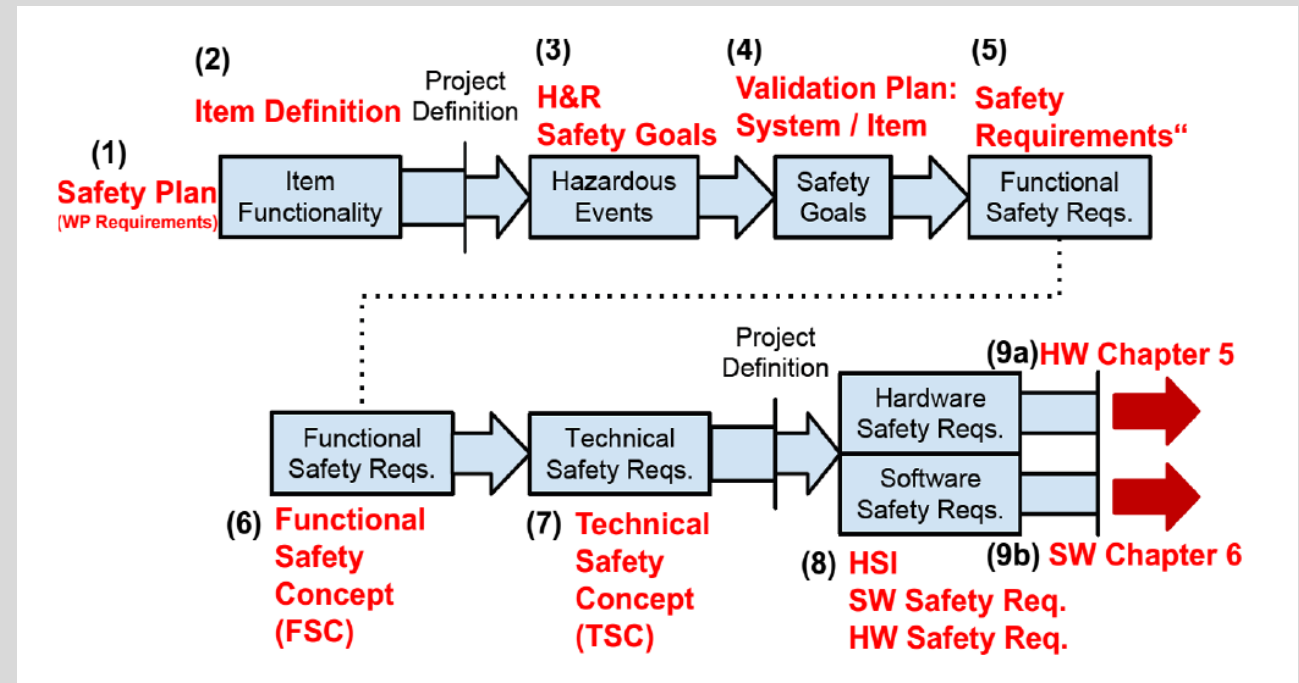
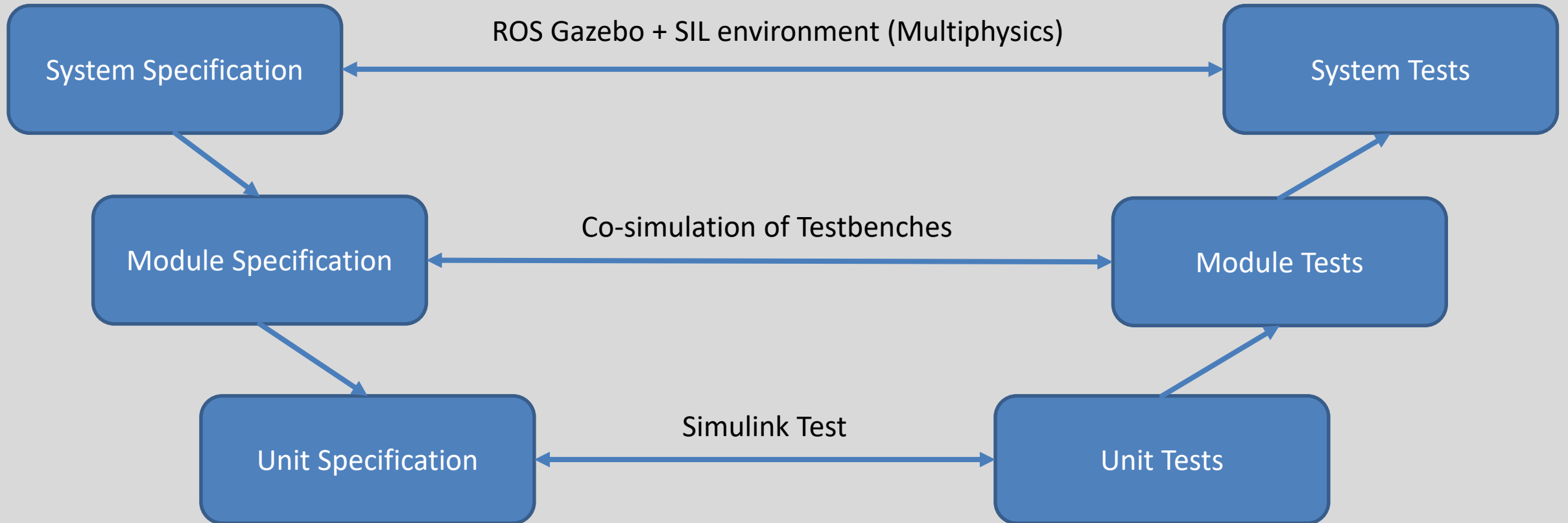


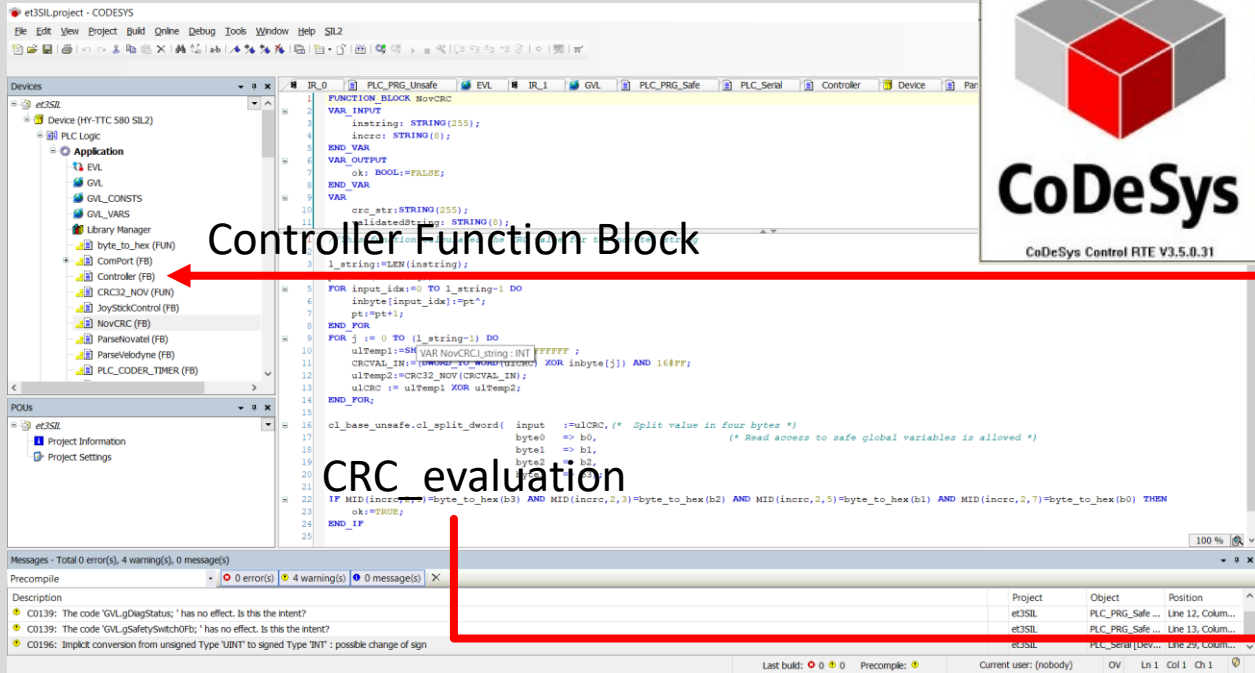
Image: ROSAS Freiburg, Paria Amini

Safety Solution



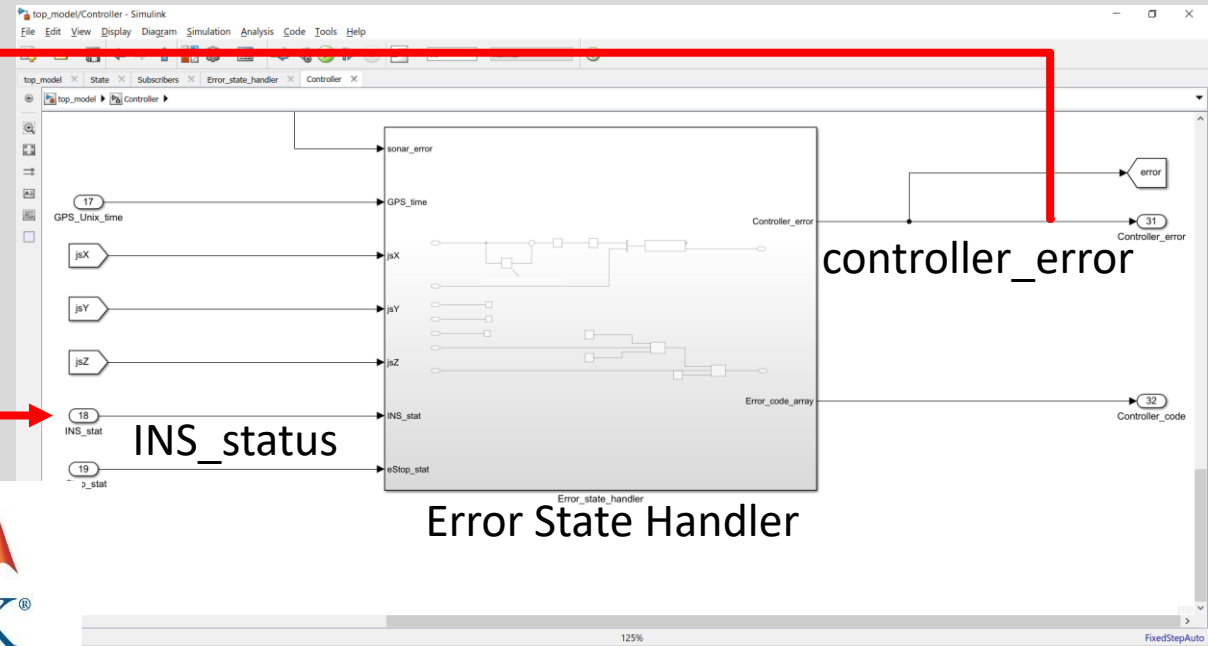
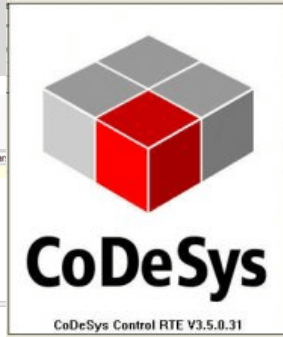
- Kyburz toolchain uses layered verification techniques and model-based design
- All requirements are easily documented for traceability

Safety Example



The screenshot shows the CoDeSys IDE with a PLC program. A red arrow points to a function block labeled "Controller Function Block". Another red arrow points to a section of code labeled "CRC evaluation". The code includes a function `NovCRC` and a `FOR` loop for CRC calculation. A third red arrow points to the "Messages" window at the bottom, which shows several warnings related to code that has no effect.

```
FUNCTION_BLOCK NovCRC
VAR_INPUT
  instrng: STRING(255);
  incre: STRING(8);
END_VAR
VAR_OUTPUT
  ok: BOOL:=FALSE;
END_VAR
VAR
  crc_str:STRING(255);
  validatedStrng: STRING(8);
  i_string:=LEN(instrng);
  FOR input_idx:=0 TO i_string-1 DO
    inbyte[instrng[incre+input_idx]]:=pt;
    pt:=pt+1;
  END_FOR
  FOR j := 0 TO (i_string-1) DO
    ulTemp1:=SH[VAR NovCRC1_string:INT FFFFFFF;
    CRCVAL_IN:=powerof2*wordofuintmem XOR inbyte[j]] AND 16#FF;
    ulTemp2:=CRC32_NOV(CRCVAL_IN);
    ulCRC := ulTemp1 XOR ulTemp2;
  END_FOR;
  c1_base_unsafe.c1_split_dword input :=ulCRC, (* Split value in four bytes *)
  byte0 => b0,
  byte1 => b1,
  byte2 => b2,
  byte3 => b3;
  IF MID(incre,1,4)=byte_to_hex(b3) AND MID(incre,2,3)=byte_to_hex(b2) AND MID(incre,2,7)=byte_to_hex(b1) AND MID(incre,2,7)=byte_to_hex(b0) THEN
    ok:=TRUE;
  END_IF
END_FUNCTION_BLOCK
```



The screenshot shows a Simulink block diagram for an "Error State Handler". It features several input ports: `INS_status` (with sub-ports `INS_stat` and `eStop_stat`), `GPS_Unix_time`, `sonar_error`, `GPS_time`, `jsX`, `jsY`, `jsZ`, and `INS_stat`. The diagram includes logic for detecting errors and setting a `Controller_error` signal. A red arrow from the CoDeSys IDE points to the `Controller_error` output of this block.



- Serial communication errors are detected and handled gracefully in control logic

Corner Cases

1	A	B	C	D	E	F	G	H	I	J	K	L	M
2	SIN	Subsystem	Conditions	Impacted Function	Process Step	Potential Effect	SEV (1= not severe, 10 = very severe)	Potential Causes	OCC (1 = not frequently, 10 = very frequently)	Current Process Controls	DET (1 = very detectable, 10 = impossible to detect)	RPN	Action recommend
3	Risk serial number	Which part of the vehicle is most impacted?	Under which conditions are failures most likely?	Specific aspect of the robot subsystem which is effected	What is the action being attempted?	In which ways can the step go wrong?	How severe is the effect on its surroundings and people?	What can cause the step to go wrong, i.e. how could the failure mode occur?	How frequently is the cause likely to occur?	What are the current controls in place to prevent this occurrence?	How probable is the detection of the failure mode or its cause?	Risk priority number RPN = SEV * OCC * DET	What actions can reduce occurrence or improve detection?
3	121	Mechanical	near curb navigation	castor lock	driving	road obstruction / edge fall / collision	8		8		8	512	
4			transition in to			move off desired path / navigation falls at dangerous moment (on							
5	46	Filters	in	INS drift	navigation	crossing etc)	8		9		7	504	
6	6	Sensors	Sun reflection	Camera	driving	road obstruction / edge fall / collision	10		8		6	480	
7	25	Command center	Unattentive operator	Movement	navigation	move off desired path / navigation falls at dangerous moment (on crossing etc)/collision	8		6		9	432	
8	7	Sensors	Sun reflection	IR/Sonar/Laser	driving	road obstruction / edge fall / collision	8		7		7	392	
9	38	Filters	near buildings	INS drift	navigation	move off desired path /	8		7		7	392	

$Risk (RPN) = Occurrence \times Severity \times Controllability$

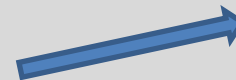
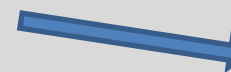


Image: drivingtests.co.nz

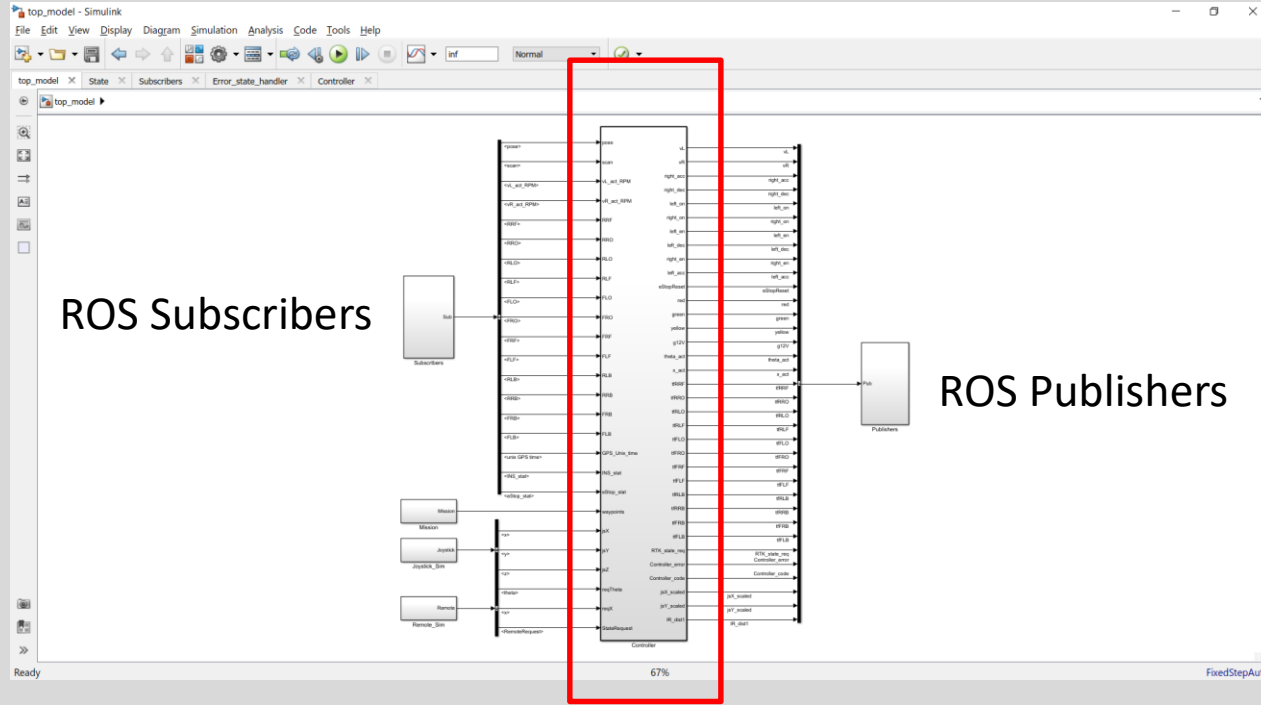


Image: arstechnica.com

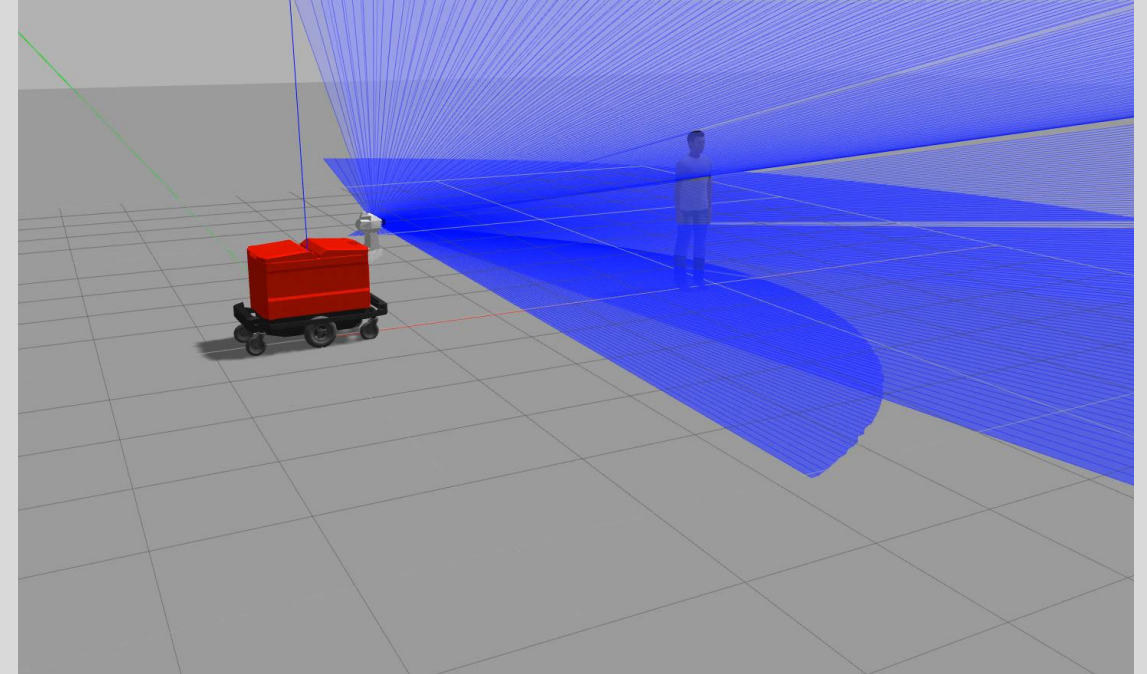


- Hazard and Risk Assessment (HARA) identified 30 failure modes with Risk Priority Number (RPN) > 200, some which are challenging to simulate

Corner Cases Solution



Autogenerated



- ROS Gazebo enables detailed sensor measurement-level simulation
- With co-simulation testing is drastically streamlined

Corner Cases Example



Summary

- Kyburz Switzerland's autonomous system developments have saved substantial development time from
 - Enabling seamless and testable control redundancy with finite state machines
 - Integrated toolboxes for streamlining development following functional safety norms
 - Simulation of difficult to test corner-cases with controller to environment interfaces

Thank you for your attention

